Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Gašper Tomažič

Zaznavanje in prepoznavanje logotipov na vozilih z računalniškim vidom

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Matej Kristan

Ljubljana, 2015

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V nalogi naslovite problem detekcije in razpoznavanja logotipov vozil v posnetkih. Na podlagi analize karakteristik logotipov načrtajte algoritem specializiran za ta razred objektov. Upoštevajte, da se v sliki lahko nahaja več vozil, da so lahko posneti na različnih razdaljah od kamere in da slike, ter s tem logotipi, niso vedno poravnani. Za namen analize algoritma pripravite in anotirajte zbirko večjega števila primerov avtomobilskih logotipov. Algoritem analizirajte po uspešnosti detekcije in razpoznavanja.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Gašper Tomažič, z vpisno številko **63100308**, sem avtor diplomskega dela z naslovom:

Zaznavanje in prepoznavanje logotipov na vozilih z računalniškim vidom

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mateja Kristana,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 14. junija 2015 Podpis avtorja:

Zahvala gre mentorju doc. dr. Mateju Kristanu za potrpežljivost, družini za potrpežljivost ter Luku, Mihu in Nini.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
MSER	maximally stable extremal regions	detektor MSER
SIFT	scale-invariant feature transform	opisnik SIFT
SURF	speeded up robust features	opisnik SURF
HOG	histogram of oriented gradients	histogram orientacij gradientov
CNN	convolutional neural network	konvolucijska nevronska mreža
ROI	region of interest	območje zanimanja
SVM	support vector machine	metoda podpornih vektorjev
HSV	hue-saturation-value color space	barvni prostor HSV
HSL	hue-saturation-lightness color space	barvni prostor HSL
MSCR	maximally stable color regions	detektor MSCR
GPU	graphics processing unit	grafična procesna enota
HD	high definition	visoka ločljivost

Povzetek

V tem diplomskem delu se posvečamo zaznavanju in prepoznavanju avtomobilskih logotipov v slikah. Kot osrednji del predlagamo postopek za detekcijo logotipov, ki temelji na odkrivanju regij MSER v sliki. Iz teh regij sestavimo kandidate za avtomobilski logotip, jih opišemo s histogramom orientacij gradientov ter z naključnimi gozdovi določimo, ali kandidatna regija predstavlja logotip, in če, kateri. Predlagali smo tudi izboljšave postopka z uporabo alternativnih barvnih prostorov, geometrijske normalizacije in uporabe lokacije registrske tablice, ki jo lahko najdemo z enakim algoritmom kot logotip. Za potrebe testiranja algoritma smo pripravili in anotirali tudi zbirko fotografij avtomobilov, ki vključuje dvajset različnih avtomobilskih znamk. Algoritem dosega več kot 70% uspešnost in z uporabo izboljšav tudi nizko število lažnih pozitivov. Čeprav se v delu osredotočimo na dokaj ozko področje, bi bilo mogoče predstavljene ideje prenesti tudi na iskanje drugih predmetov s sorodnimi lastnostmi.

Ključne besede: avtomobilski logotipi, zaznavanje, MSER, HOG.

Abstract

This thesis addresses the problem of image-based logotype detection and recognition. A new algorithm for logotype detection in images of cars is proposed. In the first stage, the algorithm localizes all maximally-stable extremal regions as candidates of logotype parts. In the next stage, the regions are combined to create logotype candidates, which are encoded by histograms of gradients. A random forest classifier is then used to verify the candidate regions as being logotypes or not and simultaneously classify them into the type of the logotype. In addition, improvements to basic algorithm are proposed. The improvements include the use of alternative color spaces, geometric normalization and use of the car license plate location to form the logotype position prior. An annotated dataset with photographs of vehicles of twenty different makes was prepared for evaluation of the algorithm. The algorithm was able to correctly localize and recognize over 70% of car logotypes at a very low false positive rate. Despite the fact that we focus on car logotype detection, the algorithm can be easily extended to detection of arbitrary logotypes or objects that do not violate assumptions we impose on the logotype appearance.

Keywords: vehicle logos, detection, MSER, HOG.

Poglavje 1

Uvod

Računalniški vid je v zadnjih letih dozorel do točke, kjer je postal zanimiv za številne aplikacije, ki so včasih potrebovale podporo človeka ali pa so bile preprosto težko izvedljive. Na primer, če bi želelo podjetje oceniti koliko časa je televizijska publika izpostavljena njihovim sporočilom prek več sto televizijskih kanalov, ali koliko oglaševanja naroča konkurenca, bi potrebovali množico ljudi, ki bi spremljali kanale, ter v športnih dogodkih, zabavnih programih in oglasih iskali omembe podjetja. Specializiran program za iskanje logotipov lahko s pomočjo računalniškega vida tako nalogo opravi bistveno ceneje in učinkoviteje.

Obstaja vrsta primerov v katerih bi bilo mogoče sistem, ki zaznava logotipe na vozilih, uporabiti. Poleg najbolj očitnih, kot je izdelava demografskih statistik ob trgovskih centrih, do bolj sofisticiranih, kot je optimizacija parkirnega prostora ali omogočanje dostopa vozilom na nujni vožnji. V kombinaciji s samodejnim branjem registrskih tablic pride v upoštev tudi za nadzor dostopa ter iskanje ukradenih vozil in vozil s premeščenimi registrskimi tablicami.

Zelo pogosta je tudi raba v sistemih za zaznavanje znamke in modela avtomobila, kjer služi za podporo ter omogoča hitrejše določanje. Če poznamo znamko je namreč mnogo lažje določiti model, saj se nabor možnih kandidatov precej skrči.

Naštete aplikacije se, v državah z od Slovenije liberalnejšim odnosom do snemanja in video nadzora, sreča tudi v praksi.

V tem delu se ukvarjamo z zaznavanjem in prepoznavanjem logotipov iz slike. Konkretno se osredotočimo na avtomobilske logotipe, vendar pa so predstavljeni postopki dovolj splošni, da bi jih bilo mogoče prenesti tudi na druge razpoznavne

simbole kot so prometni znaki.

1.1 Sorodna dela

Za potrebe industrije se je z odkrivanjem logotipov (tako avtomobilskih kot tudi ostalih) ter tudi drugih značilnih oblik ukvarjalo že precej raziskovalcev [1] [2] [3] [4]. Dokaj splošno metodo za iskanje (predvsem negibljivih, rigidnih) predmetov so predstavili Jiang, Meng in Yuan [1]. Pri tej metodi sliko večkrat razdelimo na manjše delčke. Vsakič ko je slika razdeljena, se izračuna podobnost vizualnih besed iz posameznega delčka s tistimi iz predmeta, ki ga iščemo. Če združimo te podobnosti posameznih naključnih razdelitev slike, dobimo jasno predstavo kateri deli slike so podobni našemu predmetu. Najpomembnejši prednosti tega algoritma sta gotovo možnosti paralelizacije ter izmenjave hitrosti z natančnostjo lokalizacije logotipa ali objekta vendar pa za polno učinkovitost predvideva vnaprejšnji izračun vizualnih besed iz iskalne baze.

Pričakovano največ algoritmov temelji na preizkušenem SIFTu (Scale-invariant feature transform) [5] ter njegovih izpeljankah. SIFT je način za opisovanje okolice točke v sliki. Ker okolico opiše na način, ki je neodvisen od barve, orientacije, velikosti in delno celo na izvenravninski odmik, je v računalniškem vidu zelo priljubljen. Med algoritme ki ga uporabljajo, lahko uvrstimo algoritma za detekcijo logotipov iz [2] in [3]. Prvi lokalizira logotip s pomočjo položaja registrske tablice in postopka PCFM (phase congruency feature map) [6]. Logotip se nato opiše z opisniki SIFT ter primerja z modelom s pomočjo najbližjih sosedov (angl. nearest neighbours). Nazadnje se še izvede geometrijsko validacijo s pomočjo generaliziranega Houghovega transforma (angl. generalized Hough transform) [7]. Ti postopki so običajno učinkoviti, vendar nas računska zahtevnost navadno stane možnosti procesiranja v realnem času.

V zadnjem času postajajo vedno bolj popularne konvolucijske nevronske mreže [8] [9] (angl. Convolutional Neural Network, CNN). V [10] je predstavljen algoritem, kjer se nevronska mreža uporabi za določanje vpadljivih delov slike, kjer pričakujemo logotip. Izbrano okolico opišemo s histogramom gradientov (angl. Histogram of Oriented Gradients, HOG) [11]. Ta nam omogoča, da zanimiv del slike, ki ga je našla nevronska mreža, opišemo na način, ki je neodvisen od svetlob-

nih pogojev, velikosti ter majhnih rotacij. Nato z metodo podpornih vektorjev določimo ali je izbrani del slike logotip. Omenjeni postopek je v marsikaterem pogledu podoben našemu, ki je predstavljen v tretjem poglavju, vendar se veliko bolj zanaša na algoritem MSER [12] in ne kombinira posameznih regij.

V [4] je predstavljen algoritem za zaznavo in prepoznavo prometnih znakov iz videa. Za izbiro kandidatov uporablja MSER [12], za uvrščanje pa HOG [11] in naključne gozdove (angl. random forests) [13]. Z algoritmom MSER je mogoče najti temna območja na svetlem ozadju in svetla območja na temnem ozadju, ki se običajno pojavljajo na delih slike, kjer se nahajajo objekti. Ta algoritem je najbolj soroden objavljen postopek našemu postopku, saj ta uporablja povsem enake elemente, razlikuje pa se le v podrobnostih.

1.2 Prispevki

V okviru tega dela smo sestavili algoritem, ki v slikah zaznava in prepoznava logotipe. Izdelali smo analizo njegovega delovanja in predstavili njegove prednosti in slabosti. Ker v času izdelave diplomske naloge ni obstajala obsežna prosto dostopna zbirka slik avtomobilov, primernih za naš problem, smo zgradili svojo. Sestavljena je iz slik, ki so posnete pod podobnimi pogledi, kot bi jih videle običajne nadzorne kamere. Zbirka je dostopna na spletnih straneh ViCoS [14].

1.3 Struktura naloge

Preostanek te diplome je razdeljen v štiri poglavja. V Poglavju 2 se poda osnove računalniškega vida in ozadje algoritmov predstavljenih v nadaljevanju. To poglavje razen nekaj osnovnega računalništva ne predvideva posebnega znanja. Predstavi se delovanje opisnika slik HOG [11] ter algoritem MSER [12], ki je glavni člen našega postopka za zaznavanje logotipov. V Poglavju 3 je predstavljen naš postopek za zaznavanje logotipov na avtomobilih s slik ter nekaj načinov kako to zaznavanje še izboljšati v določenih pogojih. Četrto poglavje predstavi izdelano podatkovno zbirko fotografij avtomobilov. V tem poglavju je tudi ovrednostena uspešnost našega algoritma na zbirki slik, ki smo jo pripravili ter predstavljene prednosti in slabosti algoritma. V petem poglavju predstavimo sklepe in smernice

za nadaljnje delo.

Poglavje 2

Zaznavanje in prepoznavanje v slikah

2.1 Zgradba slike

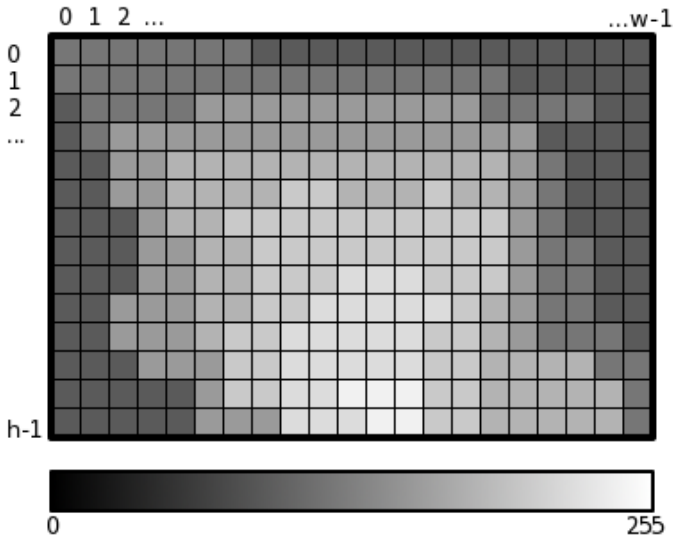
Osrednja točka računalniškega vida je slika. Ker gre za splošno razširjen izraz, si ga bomo na tem mestu, predvsem za lažje razumevanje, malo prikrojili. Naša slika I, ilustrirana na Sliki 2.1, je predstavljena z poljem, dimenzij $H \times W$ (višina $\times$ širina).

Vsaka točka v sliki (lahko tudi slikovni element ali piksel, angl. pixel) je enolično določena s parom koordinat (x, y). Koordinata x lahko zavzame vrednosti med 0 in $W - 1$ in predstavlja stolpec elementa. Koordinata y lahko zavzame vrednosti med 0 in $H - 1$ in prestavlja vrstico elementa. Koordinatno izhodišče je v zgornjem levem kotu.

Slika je sivinska (črno-bela), z 256 odtenki sive, pri čemer je 0 najtemnejša (črna), 255 najsvetlejša (bela), ostali sivinski nivoji pa so enakomerno razporejeni med njima. Vsi elementi imajo torej celoštevilsko vrednost od vključno 0 do vključno 255. Opisani postopki predpostavljajo sivinsko sliko, kljub temu, da so barvne slike veliko bolj pogoste. Te je namreč mogoče poenostaviti v sivinske, kar je pogosto prvi korak pred procesiranjem slik. S temi pretvorbami se tu ne bomo posebej ukvarjali, lahko pa za lažjo predstavo lahko omenimo, da je sivinska slika ponavadi svetlostna predstavitev barvne (torej kako svetle so posamezne točke,

ne glede na to kakšne barve so). Nekaj zgoščenih podatkov o barvah in barvnih modelih je na voljo v [15].

Intenziteto (sivinski nivo) piksla na koordinatah (x, y) v sliki I dobimo z $I(x, y)$. Na nekaterih mestih bo ta zapis poenostavljen v $I(p)$, kjer so p koordinate v sliki.



Slika 2.1: Ilustracija slike I.

2.2 Opis slike z robovi

Opisovanje slike z intenzitetami je precej podrobno, zaradi česar je težko primerjati objekte na različnih slikah. Vsakršne spremembe osvetljenosti, barve ali majhne deformacije pomenijo, da se intenzitete istega objekta razlikujejo iz slike v sliko. Da bi lažje prepoznali isti objekt ob različnih svetlobnih pogojih, zavržemo dejanske vrednosti intenzitet, ter obdržimo zgolj informacijo kako se te prostorsko spreminjajo (v kateri smeri je največja sprememba in kako velika je). Ker so na enobarvnih površinah razlike intenzitet enake 0, gre tu v resnici za opisovanje robov. Tovrstno nižanje nivoja podrobnosti je osnova za nekatere znane opisnike, kot so SIFT [5], SURF [16] in HOG [11].

2.2.1 Odvodi

V vsaki točki slike lahko določimo njen odvod v vodoravni (x) in navpični (y) smeri. Preprosto povedano, je to vrednost, ki nakaže, koliko se intenziteta (svetlost) slike spremeni v dani smeri. Če je intenziteta sosednje točke večja (svetlejša), je odvod pozitiven, če pa je intenziteta sosednje točke manjša od dane točke, pa je odvod negativen. Odvod v smeri x, v točki (i, j), lahko torej določimo kot

$$\frac{\delta I(i,j)}{\delta x} = I(i+1, j) - I(i, j).$$

Podobno je določen tudi odvod v y smeri kot

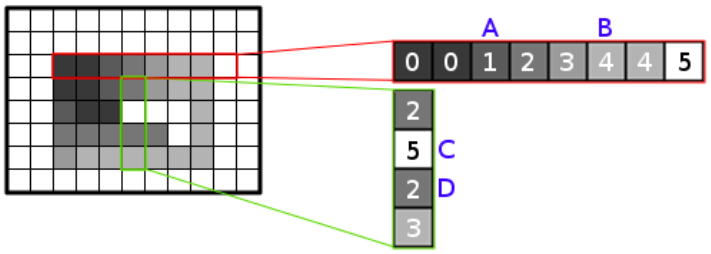
$$\frac{\delta I(i,j)}{\delta y} = I(i, j+1) - I(i, j).$$

V nadaljevanju ju bom zaradi preglednosti zapisoval kot

$$I_x(i,j) := \frac{\delta I(i,j)}{\delta x}$$

in

$$I_y(i,j) := \frac{\delta I(i,j)}{\delta y}.$$



Slika 2.2: Odvodi slike

Primer odvajanja je prikazan na Sliki 2.2, kjer sta odvoda v smeri x v točkah A in B enaka 1 in 0, odvoda v smeri y, v točkah C in D, pa -3 in 1.

2.2.2 Gradienti

Gradient ∇I v točki (x, y) je vektor

$$\nabla I(x, y) := [I_x(x, y), I_y(x, y)] .$$

Sam po sebi ne pove veliko, če pa izračunamo njegovo smer in velikost, dobimo informacijo o spreminjanju intenzitete v njegovi okolici. *Smer gradienta* v točki (x, y) je

$$\theta(x, y) := \begin{cases} \arctan\left(I_y(x, y)/I_x(x, y)\right), & I_x(x, y) \neq 0, \\ \frac{\Pi}{2}, & I_x(x, y) = 0, I_y(x, y) > 0, \\ \frac{3\Pi}{2}, & I_x(x, y) = 0, I_y(x, y) < 0, \\ nedefinirano, & I_x(x, y) = 0, I_y(x, y) = 0. \end{cases}$$

Smer gradienta kaže v smer spreminjanja intenzitete od temne proti svetli. Pri tem velja, da ne kaže nujno v vertikalni ali horizontalni smeri, temveč lahko tudi v druge smeri, odvisno od razmerja sprememb v navpični in vodoravni smeri (glej Sliko 2.3). V točkah, ki imajo enako intenziteto kot njene sosede, gradient nima (definirane) smeri.

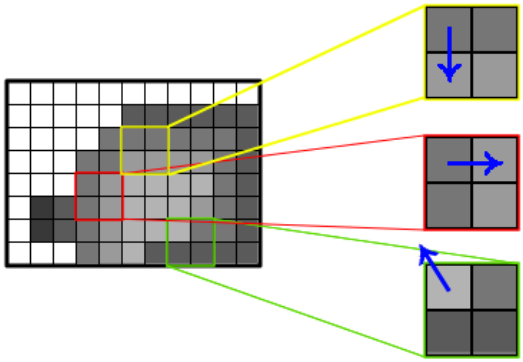
Poleg smeri lahko gradientu določimo tudi velikost kot

$$|\nabla I(x, y)| = \sqrt{I_x(x, y)^2 + I_y(x, y)^2}.$$

Ta pove, kako velika je sprememba intenzitete v sosednjih točkah. Če se sosednja polja zelo razlikujejo je lahko precej velika (pri naši sliki je to $|\nabla I(x, y)|_{max} = \sqrt{255^2 + 255^2} \approx 360.6$), če pa sta sosednji polji enaki, pa je velikost gradienta 0.

Gradient se uporablja pri vrsti problemov. Pogosto ga naprimer najdemo v algoritmih za iskanje robov v sliki. Za njih je namreč značilno, da se intenziteta na majhni razdalji precej spremeni, kar se odraža v visokih vrednostih velikosti gradienta $|\nabla I(x, y)|$. Primer takega algoritma je Cannyev detektor robov [17].

Poleg zaznavanja robov se gradiente pogosto uporablja za opisovanje slik ali delov slik. Intenzitete na sliki so namreč odvisne od osvetljenosti (v primeru barvnih slik pa tudi od obarvanosti), če pa jih opišemo z gradienti, pa ta igra veliko manjšo vlogo. Gradient namreč ohrani le informacijo o tem kako se intenziteta spreminja. Če sliko enakomerno potemnimo ali osvetlimo, so gradienti še vedno enaki. Kot je prikazano na Sliki 2.4, je primerjava dveh slik z gradienti zaradi tega veliko bolj robustna na spremembe svetlobnih pogojev.



Slika 2.3: Gradienti v sliki. Z modro puščico je prikazana smer gradienta.

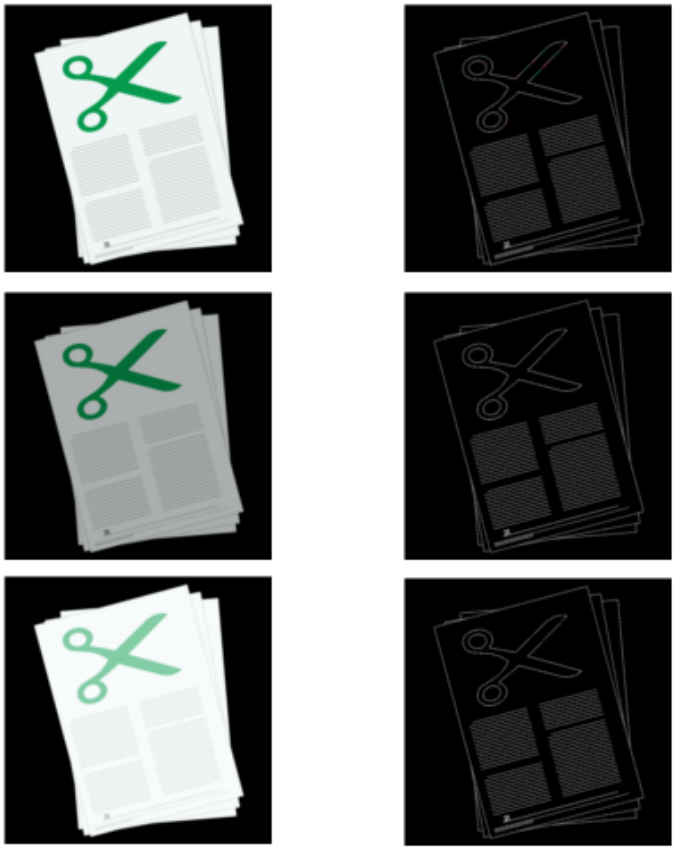
2.2.3 Histogram gradientov

V tem podpoglavju bomo opisali algoritem HOG [11], ki opis slike še izboljša. Če imamo dve sliki na katerih je isti predmet nekoliko premaknjen ali zavrten, se gradienti teh slik ne bi skladali in težko bi samo s primerjavo gradientov lahko rekli, da gre za isti predmet. Rešitev je sledeča: namesto primerjav gradientov na istih mestih v sliki izdelajmo histogram pogostosti posameznih smeri v sliki. Sliki, ki se razlikujeta samo v tem, da je predmet na njiju nekoliko zamaknjen, bosta vsebovali enako pogostost smeri in nastala histograma bosta popolnoma enaka. V primeru pa, da bi bile smeri v histogramu postavljene dovolj redko pa bi dobili še robustnost na majhne rotacije.

Izdelava histograma gradientov

Izračunamo gradiente v sliki ter jih zaokrožimo na nek izbran razmik (npr. 20° ali 35°). Nato jih utežimo z velikostjo, tako da veliki gradienti prispevajo v histogram več kot majhni. Ne gre se torej zgolj za pogostost posameznih smeri, temveč tudi njihove velikosti. Rezultat je histogram, ki predstavlja kako značilne so posamezne smeri v sliki. Na ta način bi se slika kvadrata izrazila kot histogram, ki ima štiri enako velike stolpce (tiste, ki vključujejo 0°, 90°, 180° in 270°), ostali stolpci pa bi bili 0. Ne glede na to, kako velik bi bil kvadrat in ne glede na to, kje v sliki bi bil postavljen, bi histogram izgledal enako.

Iz zgoraj opisanega bi lahko prehitro sklepali, da je to zelo učinkovit način



Slika 2.4: Osvetljenost slike ne vpliva bistveno na gradiente. Prikazane so tri različne osvetlitve iste slike (levi stolpec) in velikosti gradientov (desni stolpec). S črno barvo so označene točke z nizkimi velikostmi gradientov (0), z belo pa so označeni močni gradienti. Slednji se pojavljajo predvsem na robovih, kjer se intenziteta najbolj spreminja.

za opisovanje slik. Težava nastane pri fotografijah in drugih slikah, ki vsebujejo veliko različnih gradientov. Zavrgli smo namreč podatek, kje se ti nahajajo, in si zapomnili zgolj, kako pogosto se pojavljajo. Histogram slike, kjer se na prvi predmet A nahaja zgoraj in drugačen predmet B nahaja spodaj, bo identičen histogramu, kjer sta ta dva predmeta zamenjana.

Rešitev je vmesna pot, kjer sliko razdelimo na celice (običajno okoli 20 celic), ter za vsako izmed njih izračunamo histogram gradientov, kot je bilo opisano

zgoraj. Histograme vseh celic nato enega za drugim združimo v vektor. Iz slik ki so si podobne bo po tem postopku nastal podoben vektor, zato se lahko uporabi za robustno primerjanje slik. Imenuje se *histogram gradientov* (angl. Histogram of Oriented Gradients ali Histogram of Gradient Orientations, HOG) [11].

Normalizacija histograma

Kljub temu, da nam uporaba gradientov namesto intenzitet zagotovi manjšo občutljivost na svetlobne pogoje, jo je mogoče še izboljšati. Gradienti so namreč še vedno občutljivi na kontrast, saj večji kontrast pomeni večje spremembe v intenziteti, te pa posledično večje gradiente.

Da bi zmanjšali vpliv kontrasta uporabimo lokalno normalizacijo. V preizkusih, ki sta jih izvedla Dalal in Triggs [11], se je ta izkazala za precej pomembno za dobro delovanje HOGa.

Celice v sliki združimo v prekrivajoče se skupine (angl. blocks) ter vrednosti znotraj posamezne skupine normaliziramo. Preprost način, kako združiti celice, je naprimer združevanje štirih sosednjih celic, ki tvorijo kvadrat. Za normalizacijo Dalal in Triggs predlagata tri norme, ena izmed njih je L_2 norma.

Če je $\mathbf{v}$ naš nenormaliziran vektor, sestavljen iz histogramov celic v skupini, ter je $|\mathbf{v}|_k$ k-norma tega vektorja in ϵ majhna konstanta, je L_2 norma

$$L_2norm(v) = \frac{1}{\mathbf{v}}\sqrt{|\mathbf{v}|_2^2 + \epsilon^2},$$

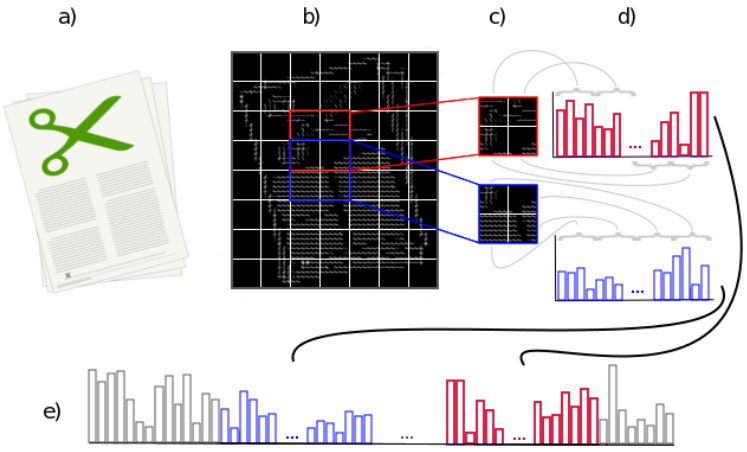
pri čemer je k-norma vektorja za pozitivno celo število k enaka

$$|\mathbf{v}|_k = \sqrt[k]{\sum_i |x_i|^k}.$$

Normalizirane vektorje skupin združimo v en sam vektor, ki predstavlja normaliziran opisnik HOG. Ker se skupine celic prekrivajo, ta vektor vsebuje histogram posamezne celice na več mestih, vsakič normaliziran v kontekstu druge skupine. Koraki izgradnje opisnika HOG so nakazani v Sliki 2.5.

2.3 Značilnice

Vrnimo se k iskanju predmetov v sliki. Da bi v sliki našli nek predmet, ga opišemo z vidnimi lastnostmi (npr. barvo ali obliko), ki so značilne posebej zanj. Od tega



Slika 2.5: Postopek HOG. a) Originalna slika. b) Gradienti slike, razdeljeni v celice. Celice so razmejene z belimi črtami. c) Skupini štirih celic, ki se na sliki prekrivata. d) Vektorja, sestavljena iz štirih histogramov, izračunanih iz celic skupin ter postavljenih skupaj. Ta vektor se normalizira. e) Vse normalizirane vektorje se združi v en skupen vektor. To je HOG opisnik slike.

kako dober bo naš opis in kako značilne bodo lastnosti s katerimi bomo predmet opisali, bo odvisna uspešnost našega iskanja.

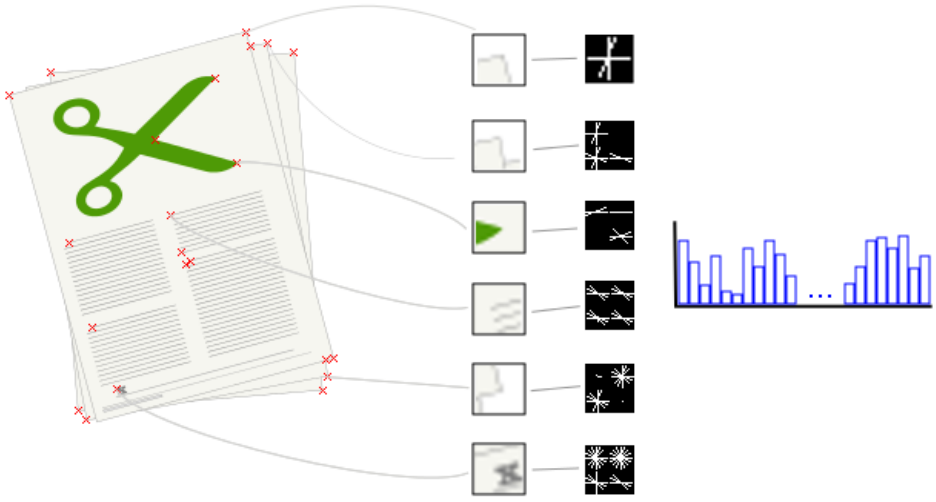
Iskanje predmeta v sliki poteka s postopkom, ki v sliki išče strukture, ki imajo iskane lastnosti. Tak postopek se imenuje *detektor značilnic* (angl. feature detector) ali preprosto *detektor*.

Ker sta barva in oblika dve precej splošni lastnosti, se v praksi pogosteje uporabi bolj podrobne elemente predmeta. Izmed teh so še posebej pomembni robovi in koti, saj dobro nakažejo mejo med predmetom in ozadjem ter jih je lahko primerjati med različnimi slikami. Za iskanje na predmetu izberemo nekaj *značilnih točk* (angl. keypoints) ter opišemo njihovo bližnjo okolico. Vsi ti opisi skupaj so predstavitev našega predmeta.

Obstaja vrsta detektorjev. Med bolj priljubljenimi je veliko takih, ki izberejo

značilne točke v kotih v kotnih strukturah na sliki ter okolico predstavijo s histogrami gradientov. Mednje spada priljubljeni SIFT (Scale-Invariant Feature Transform) ter SURF (Speeded-up Robust Features), ki je na njem osnovan. Primer takega takega detektorja je prikazan na Sliki 2.6. Omeniti velja da pri računanju histograma gradientov običajno ne gre za povsem isti postopek, kot smo ga opisali v prejšnjem poglavju, temveč katero od izvedenk. Prav tako vsak postopek definira malo drugačen način kako določiti značilne točke in kako značilnice opisati.

V podrobnosti SIFTa in SURFa se ne bomo spuščali. Več informacij je na voljo v [5], [16] in [18].



Slika 2.6: Detektor, ki za značilne točke izbere kotne strukture, ter jih opiše glede na orientacije gradientov.

Privzemimo, da je predmet opisan z eno ali več značilnicami. Da bi ga v neki novi sliki našli, iz te slike z istim postopkom izberemo značilne točke ter izračunamo značilnice. V primeru da so novo izračunane značilnice enake tistim, ki predstavljajo naš predmet, sklepamo, da smo predmet našli.

Da bi bilo to ponovno računanje čim bolj uspešno, se pri načrtovanju detektorjev upošteva določene kriterije. Mednje spadajo neobčutljivost na barvne in geometrijske spremembe v sliki, čim boljša ponovljivost (vsakokrat detektiramo

značilnico na istem delu predmeta), dovoljšne število značilnic ter programska učinkovitost (tako hitrost računanja, kot količina potrebnega pomnilnika).

2.4 MSER

2.4.1 Uvod

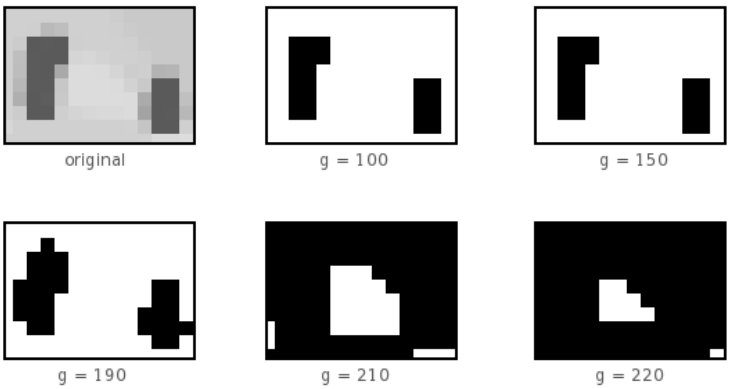
Eden izmed detektorjev značilnic, ki so dobro naslavljajo kriterije opisane v Poglavju 2.3 je MSER (Maximally Stable Extremal Regions) [12]. Sodi v kategorijo detektorjev pack (angl. blob detectors), ki imajo za značilne elemente *packe*. To so neprekinjeni deli slike, običajno nepravilne oblike in približno enakomerne intenzitete. Originalno je bil predstavljen v [12].

Packe na sliki, ki jih detektor MSER najde, so, sledeč iz imena, stabilne ekstremne regije. Imenujejo se enako kot detektor (MSER), ali MSER regije, čeprav je seveda detektor poimenovan po njih. Zelo okvirna razlaga bi bila, da so to temne lise na svetlem ozadju ali svetle lise na temnem ozadju, približno enakomerne intenzitete.

Mogoče si jih je predstavljati na sledeči način. Če v sliki vse točke, ki imajo intenziteto večjo ali enako g postavimo na 1, in vse točke ki imajo intenziteto manjšo od g na 0, dobimo binarno sliko, na kateri so vidne le dovolj svetle točke. Ta postopek se imenuje upragoavnje (angl. thresholding). Označimo sliko, ki je upragovana pri intenziteti g, z I^g.

Če bi g spreminjali od 0 do 256, bi dobili najprej čisto belo sliko (I^0), saj so vse intenzitete večje ali enake 0. Nato bi s povečevanjem g najprej temni deli slike postali črni, za njimi tudi svetli deli in nazadnje, pri $g = 256$, bi bila cela slika popolnoma črna. Nekje med procesom, bi se v sliki pojavile črne lise. Tiste lise, ki bi zaradi enakomerne svetlosti za veliko zaporednih vrednosti g ostale približno enako velike, so stabilne ekstremne regije. Na Sliki 2.7 je ilustriran ta proces.

Opisani postopek išče temne lise na svetlem ozadju, temne lise pa so le ena polovica ekstremov (minimumi). Druga polovica, maksimumi, se iščejo po istem postopku, če sliko negiramo ($\forall p \in I \rightarrow I(p) = 255 - I(p)$) ter postopek ponovno izvedemo. Običajno se pri iskanju minimumov za MSER uporablja oznaka MSER-, ter za iskanje svetlih regij na temnem ozadju MSER+. V nadaljevanju poglavja



Slika 2.7: Upragovanje slike z različnimi vrednostmi g. Prva slika je originalna slika. Na drugi in tretji sliki vidimo dve strukturi, ki se kljub velikim razlikam g ne spremenita. Ti dve strukturi sta MSERa. S povečevanjem g slika postopoma postane vedno bolj črna.

bomo govorili o MSER-, in ga bomo zapisovali kar brez minusa.

2.4.2 Definicije struktur za izračun MSER

V tem podpoglavju bomo detektor predstavili še malo bolj natančno. *Regija* v sliki Q je vsaka nepretrgana podmnožica slike I. Nepretrgana pomeni, da za poljubni točki iz regije, p in q, velja, da je mogoče priti iz p v q prek samih točk, ki so elementi Q. *Zunanja meja regije*, δQ, je množica točk, ki niso elementi regije Q, ampak imajo vsaj enega soseda v Q. Regija Q je *ekstremna regija*, če zanjo velja

$$\forall p \in Q, \forall q \in \delta Q \rightarrow I(p) < I(q),$$

torej, da so vsi elementi iz regije, manjši od vseh elementov zunanje meje regije. Tako definirana ekstremna regija je minimum. Pri MSER+ mora veljati da je $I(p) > I(q)$.

Če sliko upragujemo z naraščajočimi vrednostmi g, dobimo zaporedje slik, kjer se črne regije najprej pojavijo, nato povečujejo in združujejo ter nazadnje postanejo ena sama regija (glej Sliko 2.7). Vsakič ko g povečamo za 1, dobimo na mestu regije novo regijo, ki je po površini večja ali enaka prejšnji.

Črne regije, ki nastanejo v tem zaporedju slik lahko povežemo, tako, da sta regiji povezani, če ena vsebuje drugo. Na ta način nastane drevo, ki ima v listih ekstremne regije, ki se proti korenu združujejo in korenom, ki je ena sama velika regija. Za vsako vozlišče v drevesu (regijo), vemo pri kateri vrednosti g je bila izračunana.

Nekaterim regijam v drevesu lahko določimo stabilnost. Ta je odvisna od tega, kako hitro se spreminja velikost regije s povečevanjem g. Za regijo Q_i^g, ter regiji $Q_j^{g-\Delta}$ in $Q_j^{g+\Delta}$, ki sta za Δ oddaljeni od nje proti listu veje in proti korenu, določimo stabilnost Ψ kot

$$\Psi\left(Q_i^g\right) = \left(\left|Q_j^{g+\Delta}\right| - \left|Q_j^{g-\Delta}\right|\right) / \left|Q_i^g\right|,$$

pri čemer $|Q|$ predstavlja številnost (površino) regije Q in je Δ vhodni parameter našega postopka.

Stabilnost Ψ je odvisen od tega, kako podobni sta si regiji $Q_j^{g-\Delta}$ in $Q_j^{g+\Delta}$. Če sta popolnoma enaki je vrednost Ψ 0, kar predstavlja zelo stabilno (nespremenljivo) regijo. Regija, ki ima na poti od lista do korena najmanjšo vrednost Ψ je *maksimalno stabilna ekstremna regija.*

Včasih pa so za nas uporabne tudi regije, ki niso maksimalno stabilne, temveč zgolj dovolj stabilne ($\Psi >= \Psi_{min}$). Z dobro izbiro parametra Ψ_{min} dobimo več MSER regij, ki pa so še vedno dovolj stabilne.

2.4.3 Detekcija MSERa

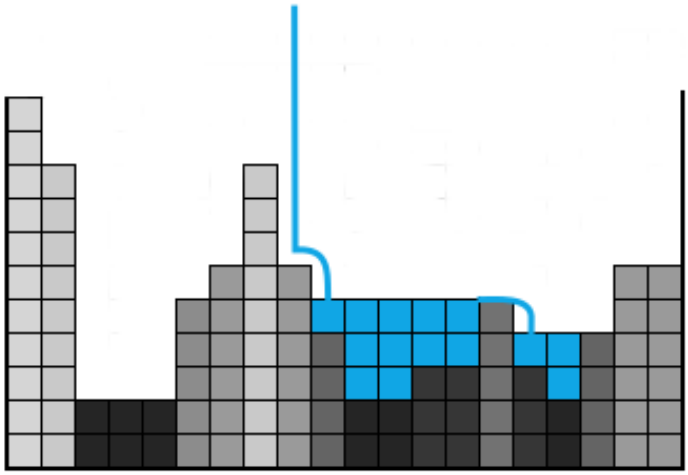
Kljub temu, da na prvi pogled deluje iskanje MSERov računsko zahtevno opravilo, so bili razviti precej učinkoviti algoritmi. Ogledali si bomo trenutno najbolj učinkovitega izmed njih.

Že v originalnem članku, [12], je bil predlagan algoritem s kompleksnostjo $O\left(n \log\left(\log\left(n\right)\right)\right)$, pri čemer je n število slikovnih elementov. Deluje s pomočjo sortiranja, ki ga je zaradi omejenega nabora vrednosti mogoče izvesti v linearnem času, ter union-find algoritma za združevanje regij.

V [19] je prikazan način za izgradnjo drevesa komponent (angl. component tree), ki se izkaže uporabno za sledenje (angl. tracking). Predstavljen algoritem je še nekoliko hitrejši od originalnega in deluje v skoraj linearnem času.

Najhitrejši poznani algoritem za računanje MSER regij pa sta predstavila Nistér in Stewénius v [20]. Ta deluje v pravem linearnem času in ima od svojih predhodnikov tudi boljše praktične lastnosti (prostorsko lokalnost za boljšo izkoriščenost predpomnilnika ter manjšo porabo glavnega pomnilnika).

Njegovo delovanje je precej naravno. Če si sliko predstavljamo kot dno neravnega bazena, kjer so temnejši deli bolj globoki in svetlejši deli slike bolj plitki, je delovanje algoritma podobno polnjenju tega bazena. Vodo na poljubni točki začnemo natakati v bazen in ta se spušča proti lokalnemu minimumu (ekstremni regiji). Ko enkrat ne more nižje, začne polniti vdolbino v katero je pritekla ter se nato prelije v druge dele bazena. Primerjava s polnjenjem je prikazana na Sliki 2.8.



Slika 2.8: Vrstica iz slike. Algoritem se iz začetne točke spusti v najnižjo dosegljivo točko. Ko ne more nižje, začne polniti bazenček, dokler ne pride do roba in se prelije v sosednjega. Spuščanje in polnjenje se potem ponavljata, dokler celoten bazen ni napolnjen.

Analogija s polnjenjem bazena se dokaj očitno izrazi v implementaciji algoritma. Za spuščanje vode v globino uporabimo prioritetno vrsto (imenovali jo bomo *priority_queue*). V njen hranimo piksle, ki jih je potrebno še raziskati. Pomembno je, da imajo višjo prioriteto piksli z nižjo intenziteto. V nekaterih pro-

gramskih jezikih bi to pomenilo, da je treba za prioriteto v vrsti podati -intenziteto. Zanko, s katero iščemo minimalno regijo najdemo v vrstici 9. Pri tem si pomagamo s funkcijo NextAccesable (glej Algoritem 2), ki pregleda sosede trenutnega piksla. Če ima kateri izmed njih nižjo prioriteto, ga nemudoma vrne za nadaljnje iskanje. Tiste pa, ki imajo višjo ali enako prioriteto, doda v vrsto za kasnejše preiskovanje. Ko je vrsta prazna, je postopek končan.

Drugi sestavni del algoritma je polnjenje bazena. Ko smo našli najnižjo (lokalno) dosegljivo točko, imamo na skladu komponente z regijami, ki smo jih našli pri iskanju minimuma. Te so na skladu razvrščene od visokih intenzitet na dnu, do najmanjše najdene intenzitete na vrhu sklada. Posamezen nivo ima lahko na skladu največ eno komponento, zato je lahko naenkrat na njem največ 255 komponent. Vsaka izmed njih vsebuje seznam pikslov v regiji, intenziteto ter zgodovino združevanja.

V proceduri ProcessStack (glej Algoritem 3) združujemo vrhnji dve komponenti s sklada, dokler ne pridemo do intenzitete, ki še ni v celoti raziskana ($next\_pixel.grey_level$). Komponenta, ki nastane z združitvijo, je sestavljena iz pikslov obeh starih komponent ter obeh njunih zgodovin. Če sta intenziteti dovolj narazen je mogoče izračunati še stabilnost regije Ψ, sicer ta izračun počaka na naslednje združevanje. Seveda je za to potrebno hraniti podatke o zgodovini komponent, njihovih velikosti in intenzitet.

Algoritem 1 Postopek za računanje MSER

```
1: global image
2: global mask ← boolean[image.width][image.height]
3: global priority_queue ← new PriorityQueue
4: global stack ← new Stack
5: global current_pixel ← image[0][0]
6: stack.push(new Component(current_pixel.gey_level))
7:
8: loop
9:     while (temp = NextAccesable(current_pixel)) != null do
10:        priority_queue.enqueue(current_pixel)
11:        current_pixel ← temp
12:        stack.push(new Component(temp.grey_level))
13:
14:    top_component ← stack.pop()
15:    top_component.append(current_pixel)
16:    stack.push(top_component)
17:
18:    next_pixel ← priority_queue.dequeue()
19:
20:    if next_pixel = null then
21:        stop
22:
23:    if next_pixel.grey_level = current_pixel.grey_level then
24:        current_pixel ← next_pixel
25:        continue
26:
27:    current_pixel ← next_pixel
28:    ProcessStack(next_pixel)
```

Algoritem 2 Funkcija, ki pregleda sosednje točke podanega piksla. Če najde točko z nižjo intenziteto, jo nemudoma vrne v obdelavo, točke z višjo ali enako intenziteto pa shrani v prioritetno vrsto. Če podan piksel nima nobenega soseda, ki bi ga bilo potrebno pregledati, funkcija vrne *null*.

```
1: function NextAccesable(pixel)
2:     for all neighbor in pixel.neighbors() do
3:         if mask[neighbor.x][neighbor.y] then
4:             mask[neighbor.x][neighbor.y] ← False
5:             if neighbor.grey_level < current_pixel.grey_level then
6:                 return neighbor
7:             else
8:                 priority_queue.queue(neighbor)
9:
10:    return null
```

Algoritem 3 Procedura, ki sprocesira regije na skladu. Regije združuje, dolker so njihove intenzitete nižje od najnižje neraziskane.

```
1: procedure ProcessStack(next_pixel)
2:     repeat
3:         top_component ← stack.pop()
4:         second_component = stack.pop()
5:
6:         if next_pixel.grey_level < second_component.grey_level then
7:             top_component.set_grey_level(next_pixel.grey_level)
8:             stack.push(second_component)
9:             stack.push(top_component)
10:            return
11:
12:        merged_component ← top_component.merge(second_component)
13:        stack.push(merged_component)
14:    until merged_component.grey_level < next_pixel.grey_level
```

Poglavje 3

Zaznavanje in prepoznavanje avtomobilskih logotipov

Iskanje in prepoznavanje logotipov je poenostavljen problem iskanja objektov v slikah, saj doda nekaj predpostavk o tem kaj iščemo. Logotipi iste vrste so si med sabo običajno zelo podobni, saj je prav prepoznavnost eden izmed njihovih namenov. Poleg tega so ponavadi preprostih oblik ter z omejenim naborom barv. Če pa upoštevamo še to, da gre za avtomobilske logotipe, pa vemo tudi to, da so v večini primerov postavljeni pokončno ter so rigidni in se ne ukrivljajo v poljubne oblike.

V tem poglavju najprej predlagamo metodo za zaznavanje in prepoznavanje avtomobilskih logotipov, ki te lastnosti uporablja v svojo prid, zatem pa še nekaj izboljšav, ki lahko povečajo učinkovitost ali hitrost samega postopka.

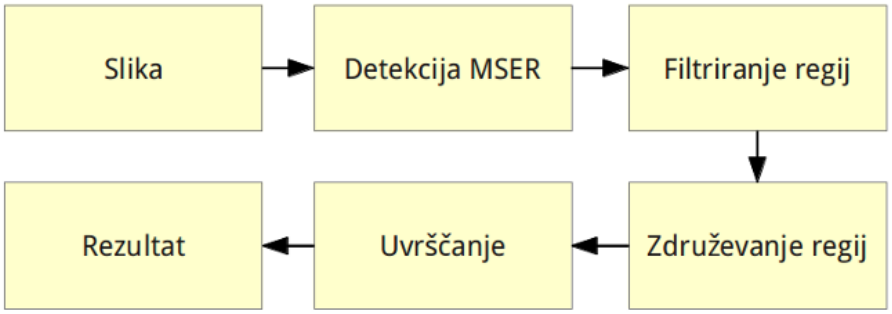
3.1 Predlagan postopek detekcije logotipov

Postopek deluje tako, da s pomočjo algoritma MSER tvori vrsto hipotez, kje bi se logotip lahko nahajal in kakšno površino bi lahko obsegal. Te hipoteze opiše s histogrami gradientov in dobljene histograme primerja s poznanimi histogrami logotipov. Če sta si histograma regije in poznanega algoritma dovolj podobna, lahko sklepamo, da je v izbrani regiji naš logotip.

Ključna razlika tega postopka od običajne uporabe detektorjev značilnic, kot

sta SIFT [5] in SURF [16], je v tem, da logotipa ne prepoznamo kot množice posameznih značilnic. Območja MSER uporabimo namreč zgolj zato, da sestavimo regije zanimanja (angl. region of interest, ROI), kjer predvidevamo, da se logotip nahaja. Če v sliki najdemo več vizualno blizu postavljenih, ali celo prekrivajočih regij, jih smatramo kot eno območje zanimanja.

Postopek za detekcijo logotipov je ilustriran na Sliki 3.1, njegovi koraki pa so podrobneje razloženi v naslednjih poglavjih.



Slika 3.1: Algoritem za zaznavanje in prepoznavanje logotipov.

3.2 Regije MSER

Prvi in tudi računsko najzahtevnejši korak je izračun MSER regij. Na njih bomo gradili regije zanimanja, kjer bomo iskali logotipe, zato jih želimo zaznati na delih logotipov ali kar na logotipih samih.

Temu primerno prilagodimo parametre MSER algoritma tako, da dovolimo dokaj nizke stabilnosti regij. To bo povzročilo, da bomo našli karseda veliko število regij. Te bomo nato izločali na podlagi njihove velikosti.

Glede omejitve velikosti najdenih regij nam je posebej pomembno, da izločimo prevelike regije, saj bi bilo območje območje zanimanja okoli logotipa lahko preveliko in ga težko klasificirali kot logotip. Premajhne regije po drugi strani ne predstavljajo takšnega problema, vendar lahko pomenijo nepotrebno računsko delo. Kako velike oziroma majhne regije izločimo je seveda povsem odvisno od velikosti slike ter seveda kako velike logotipe pričakujemo na njej.

Poleg premajhnih in prevelikih regij se iz praktičnih razlogov splača izločiti tudi preveč razpotegnjene (glej Sliko 3.2). Te namreč v okolju običajno ne predstavljajo predmetov in njihovih delov, temveč meje med predmeti in robove. Take regije je mogoče prepoznati po tem, da imajo veliko razmerje med stranicama najmanjšega očrtanega pravokotnika ali osema očrtane elipse.

Slika 3.2: Na levi sliki so z rdečo orisane vse najdene regije MSER. Na desni sliki so izločene preveč razpotegnjene regije. Regije so sicer prikazane na barvni sliki, vendar so bile izračunane na njeni sivinski različici.

3.3 Območja zanimanja

Okoli najdenih regij MSER nato določimo območja zanimanja. Vsako tako območje predstavlja kandidata za logotip. Najpreprostejši način da to dosežemo je kar pravokotnik očrtan okoli izbrane regije. Če imamo več prekrivajočih se regij, njihovo površino združimo, ter pravokotnik orišemo okoli združene regije.

Logotipi so pogosto sestavljeni iz več regij, ki jih zaznamo kot posamezne MSERe. Da bi v območje zanimanja ulovili vse dele logotipa, lahko regije MSER, ki so si dovolj blizu povežemo. To storimo tako, da postopoma sestavljamo vedno večje skupine regij in jih sproti dodajamo na seznam kandidatov za območja zanimanja. Da bi pa dobili zgolj smiselne kandidate, postavimo nekaj kriterijev, ki morajo biti izpolnjeni, da regijo dodamo in nadaljujemo rast. Kot smo že omenili bi radi, da sta si regiji dovolj blizu (upoštevamo naprimer razdaljo med najbližjima točkama regij). Poleg tega poskrbimo, da skupina regij MSER ne zavzema preve-

like površine. Za ta namen se lahko poslužimo tudi parametra, ki smo ga podali algoritmu MSER. Povečevanje lahko omejimo še s kakšnimi drugimi lastnostmi, kot je razpotegnjenost območja.

3.4 Razvrščanje regij

Ko imamo izbrana območja zanimanja lahko preverimo, če katero predstavlja logotip. Območje najprej opišemo s histogramom gradientov HOG. Nastali histogram se nato uporabi pri določanju, če je na sliki logotip in če, kateri. Ker ima nastali histogram običajno zelo obvladljivo število vrednosti (npr. 64 ali 128), lahko v ta namen uporabimo katerega od običajnih algoritmov za uvrščanje, kot so klasifikacijska drevesa [13], naključni gozdovi [21], umetne nevronske mreže [21] in metoda podpornih vektorjev (ang. support vector machine, SVM) [21].

Naključni gozdovi (angl. random forests) [13] omogočajo večrazredno uvrščanje ter oceno gotovosti odgovora, zaradi česar so zelo primerni za tako nalogo. Verjetnost posameznega razreda se oceni iz števila dreves, ki glasujejo za ta razred. Ko odločitvena drevesa v naključnem gozdu vrnejo odločitve se lahko iz njihovih glasov naredi oceno, kako verjetno je, da je večinski razred res pravi. Naš algoritem najden logotip razglasi šele, ko je ta vrednost večja od določene meje.

Ko uvrščamo regijo v razrede, moramo pri logotipih računati tudi na to, da regija mogoče sploh ne sodi v noben razred. Pri naključnih gozdovih si moramo zato izbrati minimalno gotovost, pri kateri neko regijo deklariramo kot logotip. Izbira minimalne gotovosti vpliva na to, koliko delov ozadja bomo napačno prepoznali kot logotip, ter koliko pravih logotipov bomo izpustili zaradi premajhne verjetnosti.

3.5 Izboljšave detektorja

Opisani koraki so le osnovna ideja našega algoritma. Odvisno od potreb in uporabe, pa ga lahko tudi izboljšamo z nekaterimi dodatki, ki jih opišemo v nadaljevanju.

Slika 3.3: Na levi sliki so kot pravokotniki označene regije zanimanja. Njihova barva je odvisna od verjetnosti da predstavljajo logotip pri čemer si te sledijo od modre (nizka verjetnost da gre za logotip) do rdeče (velika verjetnost). Na desni sliki je označena regija zanimanja, ki ima precej visoko verjetnost da prikazuje logotip.

3.5.1 Barvni prostor

S tem, ko sliko pretvorimo v sivinsko, zavržemo kar nekaj informacije o vsebini slike. Včasih pa je prav ta informacija ključna, da lahko ločimo regije na sliki. Informacijo o barvi je mogoče ohraniti s pretvorbo barvne slike v ustrezen barvni prostor. Primera takega prostora sta HSV in HSL [15].

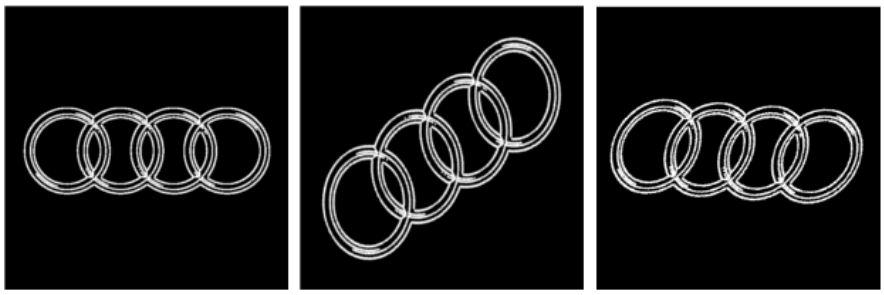
Na Sliki 3.4 so prikazani posamezni kanali slike v barvnem prostoru HSL. Ta je sestavljen iz treh komponent (kanalov), pri čemer je srednji (saturation, S) približek sivinskim slikam ki jih uporabljamo, preostala dva (hue, H in lightness, L) pa predstavljata barvo in svetlost.

Da bi v sliko vključiti še katero od preostalih komponent HSL barvnega prostora, bi po pretvorbi barvne slike v HSL postopek iskanja logotipov pognali na vsaki komponenti posebej. Pri tem bi morali biti pozorni, da bi histogram gradientov vedno računali na običajni sivinski sliki (S kanal). Alternativa temu je algoritem MSCR, predstavljen v [19], ki zaznava regije glede na njihovo barvno stabilnost.

Slika 3.4: Komponente barvnega prostora HSL. Pri običajnem postopku se uporabi samo drugi kanal (srednja slika), čeprav je lahko logotip bistveno bolje izražen na katerem od preostalih dveh. V tem primeru je sicer rumeno obarvan logotip zelo izrazit na H kanalu (leva slika).

3.5.2 Geometrijska normalizacija

Kljub temu, da si logotipe običajno predstavljamo v pokončni legi, se v splošnem na naravnih slikah lahko logotip pojavi v različnih orientacijah. Da bi dosegli čim višjo klasifikacijsko natančnost vektorjev histogramov gradientov, pa morajo biti najdeni logotipi čim bolj podobni tistim, ki smo jih uporabili pri učenju. Ker pa je v praksi težko doseči, da bi se ta vedno pojavil pod enakim kotom, se je smiselno posluževati normalizacije regije. Ta poravna različno orientirane in zamaknjene logotipe v isto postavitev.



Slika 3.5: Na levi sliki je prikazan izvoren logotip. Na sredini je isti logotip gledan z drugega zornega kota. Na desni sliki pa je normaliziran pogled sredinske slike. Normalizacija je bila tu izvedena zgolj za rotacijo.

Da bi sliko poravnali v neko osnovno lego, je potrebno najprej določiti njeno

trenutno postavitev. Pri tem nas posebej zanima kako je njena vsebina raztresena, ter v katero smer. Te informacije je mogoče pridobiti s pomočjo momentov slike.

Moment je s koordinatami utežena vsota intenzitet slike. Od njegove stopnje je odvisno, katera koordinatna os se bolj upošteva. Za stopnji a in b je moment slike podan kot

$$M_{ab} = \sum_x \sum_y x^a y^b I(x, y).$$

To, da momenti povežejo intenzitete v sliki z njihovimi pozicijami (koordinatami), nam že daje slutiti povezavo med raztrosom oziroma navideznim naklonom vsebine slike. Vmesni korak pa je kovariančna matrika. Kovariančna matrika C, sestavljena iz treh centralnih momentov (momentov popravljenih na težišče podatkov) prvega in drugega reda

$$\mu'_{20} = \frac{M_{20}}{M_{00}} - \bar{x}^2,$$

$$\mu'_{02} = \frac{M_{02}}{M_{00}} - \bar{y}^2,$$

$$\mu'_{11} = \frac{M_{11}}{M_{00}} - \bar{x}\bar{y},$$

pri čemer sta $\bar{x} = M_{10}/M_{00}$ in $\bar{y} = M_{01}/M_{00}$, koordinati navideznega težišča slike. Matrika sama je nato sestavljena kot

$$C(I) = \begin{bmatrix} \mu'_{11} & \mu'_{20} \\ \mu'_{02} & \mu'_{11} \end{bmatrix}.$$

Kovariančno matriko si lahko predstavimo kot rotirano elipso, ki se prilagaja vsebini slike in naš cilj je poravnati osi te elipse z koordinatnim izhodiščem ter tako skalirati sliko, da bosta osi enaki. V primeru da imamo več slik istega logotipa z različnimi orientacijami, bo poravnanje elips, ki se nanje prilegajo, povzročilo, da bodo vsi logotipi enako poravnani. V mislih je sicer potrebno imeti, da bodo poravnani glede na najmočnejšo smer elipse in ne nujno tako, kot smo ljudje logotipa navajeni.

Izračun velikosti in naklona osi prilegane elipse, sicer temelji na lastnih vrednostih in vektorjih kovariančne matrike, vendar ga je mogoče z nekaj manipulacije zapisati zgolj z uporabo momentov. Kot φ navidezne nagnjenosti slike se tako izračuna kot

$$\varphi(I) = \frac{1}{2} \arctan\left(\frac{2\mu'_{11}}{\mu'_{20} - \mu'_{02}}\right).$$

Ko poznamo orientacijo lahko nato izvedemo rotacijo s pomočjo rotacijske matrike ter ustrezne transformacije. Če bi hoteli sliko zavrteti okoli navideznega težišča $T(\bar{x}, \bar{y})$ pod kotom φ, zgradimo matriko

$$R = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) & (1-\cos(\varphi)) \cdot \bar{x} - \sin(\varphi) \cdot \bar{y} \\ -\sin(\varphi) & \cos(\varphi) & \sin(\varphi) \cdot \bar{x} - (1-\cos(\varphi)) \cdot \bar{y} \\ 0 & 0 & 1 \end{bmatrix}.$$

Da bi popravili še različne raztege, potrebujemo tudi matriko, ki bo te poenotila. Razteg predstavljata lastni vrednosti kovariančne matrike $C(I)$, ki ju z momenti izračunamo kot

$$\lambda_{1,2} = \frac{\mu'_{20} + \mu'_{02}}{2} \pm \frac{\sqrt{4{\mu'}_{11}^2 + (\mu'_{20} - \mu'_{02})^2}}{2}.$$

Približno enakomernost slike dosežemo s takšnim raztegom, ki izenači obe lastni vrednosti λ. V matrični obliki ga zapišemo z

$$S = \begin{bmatrix} \lambda_1/\lambda_2 & 0 & 0 \\ 0 & \lambda_2/\lambda_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Transformaciji vrtenja in raztega lahko združimo v eno, tako, da ju pomnožimo

$$P = R \cdot S.$$

Izdelana matrika se nato lahko uporabi za izvedbo transformacije. V danem primeru gre za afino (izvenravninsko) transformacijo, ki je namenjena spreminjanju pogleda na dano ravnino (sliko). Omogoča vrsto znanih geometrijskih sprememb, kot so razteg, vrtenje in translacija. Izmed teh se pri normalizaciji običajno uporabljajo vrtenje, skaliranje in premikanje, slednje zgolj za centriranje vrtenja.

S transformacijo matriko P in sliko I, ki jo želimo normalizirati, izračunamo normalizirano sliko I', tako da pikslom I' priredimo vrednosti pikslov I po sledeči formuli

$$I'(x,y) = I\left(\frac{P_{11}x + P_{12}y + P_{13}}{P_{31}x + P_{32}y + P_{33}}, \frac{P_{21}x + P_{22}y + P_{23}}{P_{11}x + P_{12}y + P_{13}}\right).$$

Nekaj več informacij o momentih je na voljo na [15].

3.5.3 Paralelizacija

Moderni računalniški sistemi prinašajo prednost z uporabo večjedrnih in grafičnih procesnih enot. Algoritem lahko tako pridobi hitrost z deljenjem dela na te podsisteme in veliko jih je tako dobilo tudi svojo različico s paralelizacijo. Mednje spada histogram gradientov za katerega so razvili fHOG, hitrejšo različico, ki uporablja grafično procesno enoto. Predstavljen je v [22]. Poleg tega obstaja podobna implementacija v priljubljeni knjižnici OpenCV [23].

Tudi naključni gozdovi imajo verzije ki uporabljajo grafično procesno enoto. Te predstavljajo v primerjavi z običajno implementacijo nekajkratno pohitritev, vendar so omejene z manjšo količino pomnilnika, ki je ponavadi na voljo pri grafičnih enotah.

Nekoliko težavneje je pri MSER. Za ta algoritem namreč ne obstaja nobena prosto dostopna izvedba s paralelizacijo. Na spletu trenutno obstaja ena omemba uporabe paralelne implementacije algoritma MSER, nakazana v [24], vendar gre za zaprto znanje zasebnega podjetja.

3.5.4 Kontekst

Namesto preiskovanja celotne slike, se lahko omejimo le na območja, na katerih pričakujemo logotip. S tem pohitrimo preiskovanje in občutno zmanjšamo možnost lažnih pozitivov.

Kako doseči zmanjšanje preiskovanega območja je odvisno od posamezne domene. Pri avtomobilskih logotipih je to območje omejeno na vidni del avtomobila (brez ozadja) ali na pas okoli registrske tablice, kjer se logotipi običajno pojavljajo. Avtomobil je na primer mogoče ločiti od ozadja če imamo posnetek, na katerem se avtomobil premika, ali če imamo sliko ozadja, ki ne vsebuje avtomobila. Primer algoritma, ki v videu loči ozadje od gibajočih delov (in celo senc), najdemo v [25].

Ker so si avtomobili v postavitvi elementov (luči, zračnikov, registrske tablice) v veliki meri precej podobni, si lahko pomagamo s to informacijo da lažje izločimo nekatere hipoteze. Če bi poznali, na primer, položaj registrske tablice, bi lahko ocenili, da je logotip nekje v bližini, najverjetneje nekje v pasu nad registrsko tablico. In ker je registrska tablica dokaj izstopajoč del avtomobila, in jo je mogoče zaznati z enakim algoritmom kot logotipe, si z njeno lokacijo na preprost način

pridobimo dodatno informacijo o potencialni lokaciji logotipa. Vse kar je potrebno storiti, je prilagoditi parametre algoritma za zaznavanje logotipa na nekoliko večje in bolj pravokotne oblike, ter naučiti ločen model za klasifikacijo hipotez registrskih tablic.

3.5.5 Izboljšan sistem za detekcijo

Naštete izboljšave se lahko uporabijo kot dopolnitev osnovnih korakov algoritma, odvisno od aplikacije. Če se iskani predmeti pojavljajo v določenih barvnih prostorih (barvne nalepke) ali če se slabo odražajo v barvnem prostoru intenzitet (sivinska slika), lahko uporabimo druge barvne prostore in njihove kanale, na primer prej predstavljen HSL. Če se naši predmeti pojavljajo pod različnimi koti, lahko izboljšamo zaznavanje z geometrijsko normalizacijo in če imamo kaj informacije o vsebini slike, jo lahko uporabimo za oženje potencialnih kandidatov za naše predmete. Marsikatere korake osnovnega algoritma je mogoče tudi paralelizirati, in s tem pohitriti njihovo delovanje.

Poglavje 4

Eksperimentalna evaluacija

Da bi preizkusili postopek opisan v prejšnjem poglavju, smo pripravili zbirko fotografij avtomobilov v različnih svetlobnih pogojih. Algoritem za zaznavanje logotipov smo implementirali v programskem jeziku Python, s pomočjo knjižnic OpenCV [23] in SciKit [26]. Izvedli smo eksperimentalno analizo našega algoritma ter preverili kako uspešno je zaznavanje in kritično analizirali algoritem.

4.1 Zbirka slik

Sestavljena zbirka vsebuje 340 fotografij avtomobilov. Zajete so v resolucijah 3648x2763 in 2272x1704 pikslov. Avtomobili so bili fotografirani z razdalj enega do dveh metrov, pod različnimi koti. Vsebujejo tako sprednje kot tudi zadnje dele avtomobila. Nekaj primerov je prikazanih na Sliki 4.1.

Slika 4.1: Primerki fotografij iz zbirke podatkov.

Pri izdelavi zbirke smo poskrbeli za zasebnost lastnikov avtomobilov. Na vseh fotografijah so bili zamegljeni obrazi naključnih mimoidočih, ter odstranjene nalepke in oznake, ki bi lahko vodile v prepoznavno avtomobila. Registrske tablice so bile zamegljene z istim postopkom kot je bil uporabljen za zaznavno logotipov in je opisan v prejšnjem poglavju. Edina potrebna sprememba so bili parametri algoritma MSER, saj so registrske tablice nekoliko večje od logotipov.

Vključenih je 20 avtomobilskih znamk, konkretno so to: Kia, Skoda, Mitsubishi, Citroen, Chevrolet, Mercedes-Benz, Fiat, Volkswagen, Seat, Audi, Honda, Nissan, Subaru, Peugeot, Renault, BMW, Ford, Opel, Volvo, Hyundai in Suzuki. Posamezna znamka vključuje pet do dvajset slik, odvisno od pogostosti avtomobila.

Na vseh slikah so bili ročno označeni logotipi. Vsaka slikovna datoteka ima tako pripadajočo tekstovno datoteko z imenom *<ime slike>.makes.txt*, ki vsebuje seznam logotipov in njihovih pozicij v sliki. Vsak logotip je naveden v svoji vrstici in obsega pet stolpcev ločenih s podpičji, ki so po vrsti:

1. Horizontalna (X) pozicija zgornjega levega kota logotipa
2. Vertikalna (Y) pozicija zgornjega levega kota logotipa
3. Širina logotipa (v pikslih)
4. Višina logotipa (v pikslih)
5. Šifra logotipa

Šifre in njihova ujemajoča imena (imena znamk) so našteta v datoteki *makes.tags.txt*, ki je priložena zbirki podatkov.

4.2 Testiranje

Za preizkus uspešnosti algoritma smo pripravili demonstracijsko implementacijo, program za testiranje, ter nekaj podpornih programov.

Program za zaznavanje logotipov pregleduje slike iz zbirke ter poskuša na njem najti logotipe. Za vsako sliko pošlje seznam najdenih logotipov (ta je lahko tudi prazen) programski kodi, ki preverja rezultate. Da se logotip šteje kot zaznan,

mora biti razmerje med presekom in unijo ploščin pravega (označenega) logotipa ter tistega, ki ga je našel algoritem, vsaj 0.3. Ocenili smo, da je to dovolj dobra meja za lokalizacijo, čeprav se pri testih izkaže, da imajo vsi razmerje nad 0.5, običajno tudi nad 0.8. Sistem sicer loči med pravilno zaznanimi logotipi, najdenimi vendar narobe razvrščenimi (napačna znamka), lažnimi zaznavami (ozadje prepoznano kot logotip) in logotipi v sliki, ki niso bili zaznani.

4.2.1 Gradnja učne množice

Testiranje smo izvajali z 10-kratnim prečnim preverjanjem (angl. k-fold cross validation). Iz logotipov v učni množici smo izpeljali več sto različic z naključnim spreminjanjem svetlosti, kontrasta, perspektive, velikosti in dodajanjem šuma. Iz te generirane množice logotipov smo zgradili naključne gozdove z različnim številom odločitvenih dreves ter z različnimi števili atributov.

Testi so bili izvedeni na testni množici z nekaj različnimi nabori parametrov, izvedbami HOG algoritma (število celic, blokov, normalizacijo), ter z in brez uporabe barvnih prostorov in normalizacije.

4.2.2 Mere uspešnosti

Za ocenjevanje uspešnosti algoritmov ter posameznih naborov parametrov smo pri izvajanju merili naslednje statistike:

1. Pravilni (pravilni pozitivi, angl. true positives) – število logotipov, ki jih je algoritem pravilno uvrstil in dovolj natančno lociral.

2. Napačni (napačni pozitivi, angl. false positives) – število delov ozadja, ki ga je algoritem zmotno prepoznal kot logotip ali logotipi, ki jih ni dovolj natančno lociral.

3. Neodkriti (napačni negativi, angl. false negatives) – število logotipov v sliki, ki jih algoritem ni zaznal.

Poleg naštetih smo v opazovali tudi pravilno locirane logotipe, vendar narobe uvrščene (napačna znamka), vendar v nobenem izmed izvedenih poizkusov nismo imeli takega primera, zato jih v grafih izpustimo. Za poljuben dejanski logotip na

sliki velja, da sodi v eno izmed skupin pravilnih, neodkritih ali narobe uvrščenih, medtem ko je napačnih lahko poljubno mnogo. Na grafih prikazujemo te statistike v razmerju s številom vseh logotipov, kar pomeni da se pravilni in neodkriti se seštejejo v 1, napačnih pa je lahko več.

Poleg naštetih smo iz njih izračunali bolj standardni meri *natančnost* (angl. precision) in *priklic* (angl. recall). Iz naših meritev ju lahko izračunamo kot

$$\text{natančnost} = \frac{\text{pravilni}}{\text{pravilni} + \text{napačni}}$$

in

$$\text{priklic} = \frac{\text{pravilni}}{\text{pravilni} + \text{neodkriti}}.$$

Iz njunih enačb je razvidno, da natančnost predstavlja delež pravilnih izmed vseh, ki smo jih zaznali, priklic pa je delež pravilnih izmed vseh logotipov, ki obstajajo v zbirki. Natančnost bo torej povedala ali algoritem mogoče prepogosto razglasi neko regijo za logotip (slepo ugibanje), priklic pa, kolikšen delež logotipov dejansko najdemo.

4.3 Rezultati

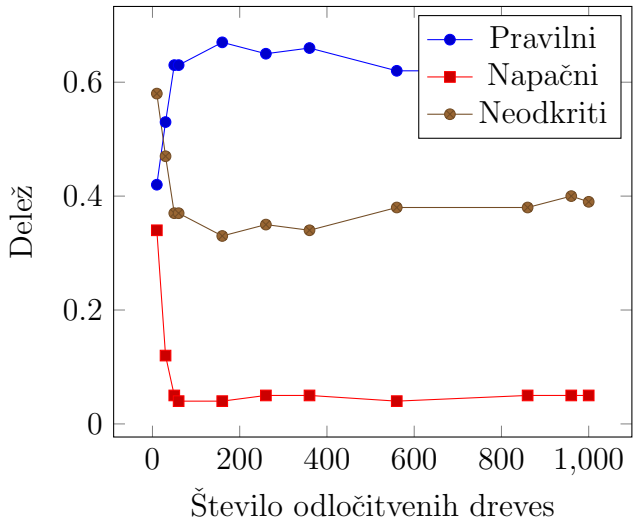
4.3.1 Vpliv števila dreves

Eden izmed parametrov, ki smo jih spreminjali, je bilo število odločitvenih dreves v modelu naključnega gozda. Privzeto število smo povečevali do 1000 z neenakomernimi razmaki. Na Sliki 4.2 vidimo, uspešnost algoritma v odvisnosti od števila dreves v naključnem gozdu. Vsako drevo je bilo zgrajeno na $\sqrt{n}$ naključno izbranih atributov. Kot mera nečistoče je bil uporabljen Gini-index [27].

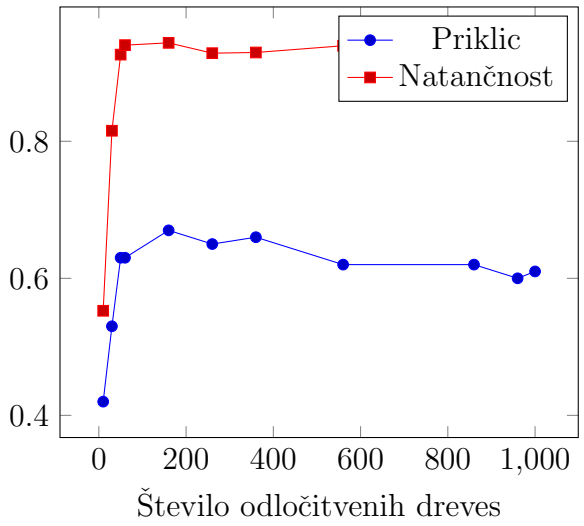
Izkaže se, da število odločitvenih dreves v modelu ne vpliva bistveno na uspešnost algoritma po vrednosti 150 in se je celo bolje držati nekoliko manjših modelov. Veliki modeli imajo tudi to slabost, da lahko zahtevajo zelo veliko količino pomnilnika, ter potrebujejo dlje da uvrstijo vektor v razred.

4.3.2 Vpliv parametrov HOG

Preizkusili smo nekaj različnih parametrov HOG. Za vsako spremembo smo na novo naučili klasifikacijski model. Na Sliki 4.4 in Sliki 4.5 so prikazani rezultati za



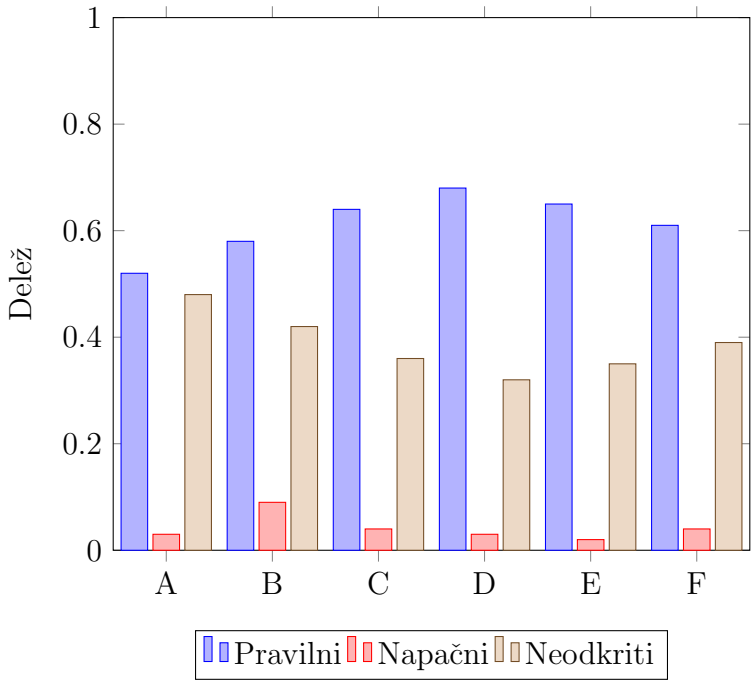
Slika 4.2: Delež pravilno zaznanih, neodkritih in napačnih logotipov v odvisnosti od števila odločitvenih dreves v modelu naključnega gozda.



Slika 4.3: Priklic in natančnost v odvisnosti od števila dreves

različne izvedbe HOG.

Ker so bila območja zelo različnih oblik, smo jih, da bi dobili enako velike vektorje, razdeljevali na fiksno število celic. Njihovo število smo ohranjali majhno, saj so daljši vektorji pomenili zelo velik odločitveni model, kar je precej otežilo testiranje. Ocenjujemo, da večje število celic (in blokov) ne bi bistveno pozitivno



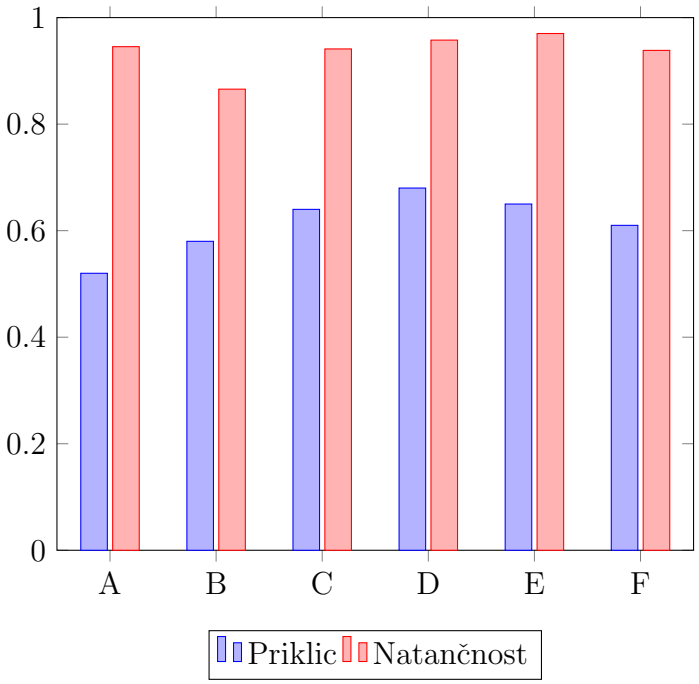
Slika 4.4: Delež pravilno zaznanih, neodkritih in napačnih logotipov za različne implementacije HOG. A) Implementacija SciKit 3×3 brez normalizacije B) implementacija SciKit 2×2 brez normalizacije C) Implementacija OpenCV 3×3 z lokalno normalizacijo D) Implementacija OpenCV z 2×3 z lokalno normalizacijo E) Implementacija OpenCV 2×2 brez normalizacije F) Implementacija OpenCV 2×3 z globalno normalizacijo

vplivalo na rezultate saj je velik delež napak prišel iz naslova sestavljanja območja zaupanja.

4.3.3 Vpliv klasifikacijske meje

Z večanjem zahtevane verjetnosti dobimo manj lažnih pozitivov, vendar tudi več pravih logotipov izpustimo, saj nismo dovolj prepričani v njihovo pravilnost. Na Sliki 4.6 in Sliki 4.7 vidimo kako se uspešnost algoritma spreminja s spreminjanjem klasifikacijske meje.

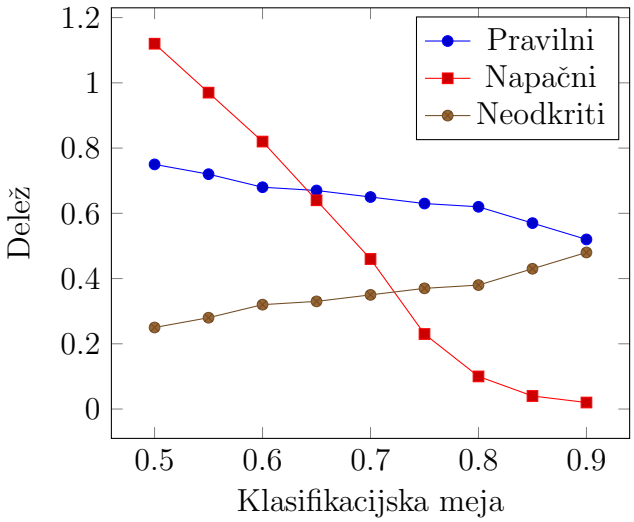
Vpliv izbire klasifikacijske meje na zgrešene zadetke je zelo velik. Pri majhnih vrednostih se v območjih zanimanja na močno teksturiranih površinah, kot



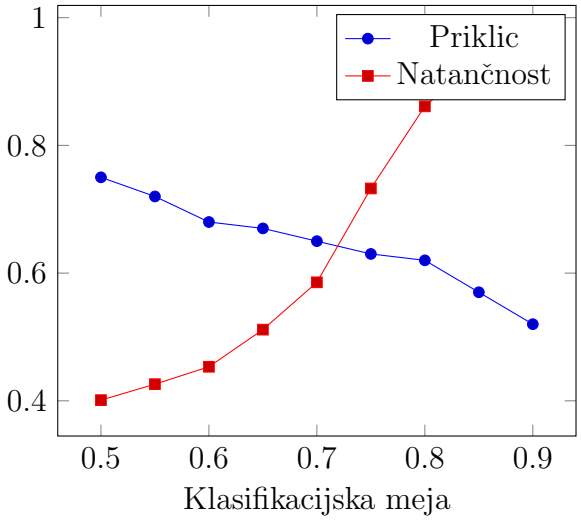
Slika 4.5: Priklic in natančnost za različne implementacije HOG. A) Implementacija SciKit 3×3 brez normalizacije B) implementacija SciKit 2×2 brez normalizacije C) Implementacija OpenCV 3×3 z lokalno normalizacijo D) Implementacija OpenCV z 2×3 z lokalno normalizacijo E) Implementacija OpenCV 2×2 brez normalizacije F) Implementacija OpenCV 2×3 z globalno normalizacijo

sta rastlinje ali pesek, pojavi veliko število lažnih pozitivov, saj so dovolj pogosto gradienti podobni tistim na logotipih. S povečevanjem meje ta učinek izginja, vendar pa se tudi logotipi, ki niso dovolj podobni učnim primerom pogosteje zavržejo kot napačni. Na Sliki 4.8 je prikazano, kako se v delih slike z rastlinjem, zaradi teksturiranosti, pojavi veliko število kandidatov za logotip.

Ker je znaten delež lažnih zadetkov zaznan v ozadju (torej ni del vozila), bi uporaba konteksta, predstavljena v prejšnjem poglavju, bistveno izboljšala prikazane rezultate.



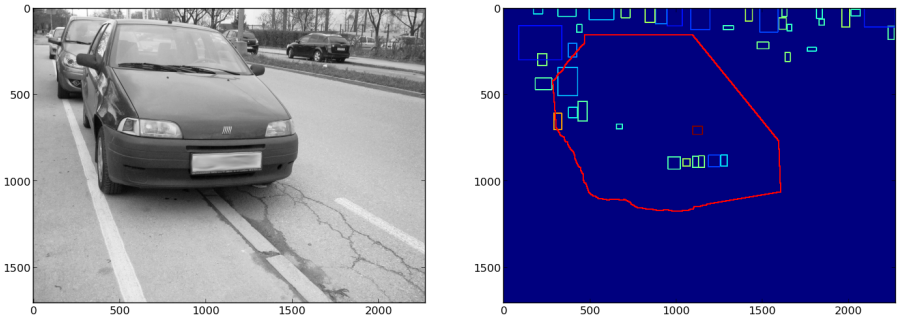
Slika 4.6: Delež pravilno zaznanih, neodkritih in napačnih logotipov v odvisnosti od klasifikacijske meje.



Slika 4.7: Priklic in natančnost v odvisnosti od klasifikacijske meje.

4.3.4 Vpliv uporabe konteksta

Ker se želimo v našem primeru osredotočiti zgolj na logotipe, ki se nahajajo na avtomobilih, je nesmiselno upoštevati hipoteze, za katere vemo, da se nahajajo izven njih. Ker je z našim algoritmom relativno lahko detektirati registrske tablice,



Slika 4.8: Na sliki je primer z visoko postavljeno klasifikacijsko mejo. Medtem ko je v območju avtomobila (označen z rdečo) le nekaj potencialnih kandidatov, jih je v ozadju precej več.

si lahko pomagamo z informacijo o lokaciji tablice, in predpostavimo, da se logotip nahaja nekje v pasu pet višin tablice nad njo. Vse hipoteze ki se ne nahajajo v tem območju izločimo.

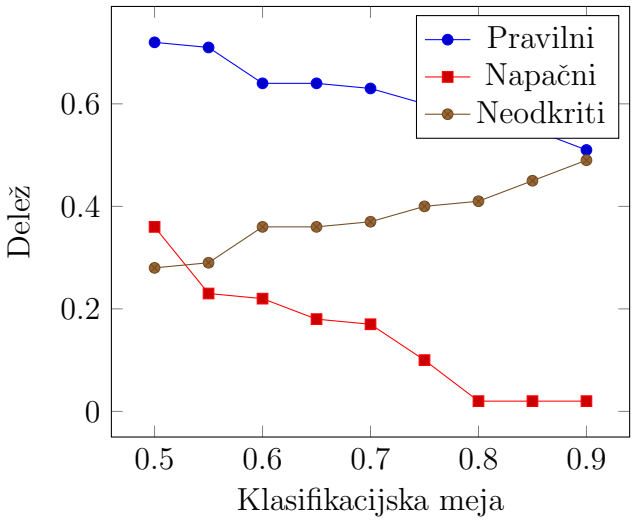
Na Sliki 4.9 so prikazani deleži pravilnih, napačnih in neodkritih logotipov. Medtem ko je pravilnih logotipov le malenkost manj, je v primerjavi s Sliko 4.6 bistveno manj lažnih pozitivov.

4.3.5 Kvalitativna evaluacija

Slika 4.10 prikazuje nekaj slik na katerih je bil pognan algoritem za detekcijo logotipov. Na fotografijah v prvi vrstici sta bila logotipa uspešno najdena. Na nobeni izmed njiju ni bilo lažnih pozitivov, kljub temu, da je, predvsem na desni, relativno veliko šuma.

Če je logotip ni dovolj izpostavljen, bodisi zaradi prenizkega kontrasta, bodisi zaradi prevelike vključenosti v druge dele avtomobila, ga algoritem ne bo našel. Na Sliki 4.10 sta v drugi vrstici prikazana taka primera. Na levi sliki je logotip vključen v masko, zato so nastale regije zelo velike in odstranjene kot neprimerne. Logotip na desni je premalo kontrasten, zato detektor MSER na njem ne zazna regije, ki bi sicer verjetno pomenila, da bi logotip našli.

Zadnja vrstica na sliki 4.10 pa prikazuje primera, kjer sta se poleg logotipov



Slika 4.9: Delež pravilno zaznanih, neodkritih in napačnih logotipov v odvisnosti od klasifikacijske meje, pri uporabi konteksta.

pojavili še dve lažni zaznavi. Te so najbolj pogoste na delih slike kjer se nahaja rastlinje, saj to povzroči še posebej veliko šuma.

4.3.6 Slabosti

Algoritem je v celoti odvisen od regij MSER, in če te na območju logotipa niso dovolj značilne (npr. zaradi premajhne variance ali velikosti) in so zaradi tega izločene. V takih primerih se zgodi, da okoli logotipa območje zanimanja sploh ne nastane, ali pa prekriva zgolj del logotipa.

Druga slaba točka pa je gotovo veliko število parametrov, ki jih je potrebno nastaviti, da algoritem dobro deluje na izbranih slikah. Tako je potrebno oceniti spodnjo in zgornjo mejo velikosti regij MSER, ki jih upoštevamo, klasifikacijsko mejo, za čim boljše razmerje med lažnimi zadetki in neodkritimi logotipi.

4.3.7 Prednosti

Ena izmed prednosti našega postopka je velika robustnost na skalo. Sposoben je zaznati od dokaj majhnih do velikih logotipov (glej Sliko 4.11). Tu sicer igra vlogo izbira parametrov, s katerimi omejimo velikosti glede na naravo slik, vendar

Slika 4.10: Na sliki so prikazani rezultati algoritma na nekaterih slikah iz zbirke. Z zelenim pravokotnikom so označene regije, ki jih algoritem prepoznal kot logotip, modri pravokotniki pa so regije, prepoznane kot ozadje.

algoritem dopušča precej razlik. V zbirki je program zaznal logotipe velikosti 50×50 pikslov in tudi take, ki so veliki 300×300 pikslov (več kot 30-krat večji).

Druga pomembna prednost je njegova hitrost, saj lahko procesira slike v realnem času. V testih izvedenih z Intelovim procesorjem i3-370M (2,4 GHz) ter relativno ne-optimiziranim programom je lahko v eni sekundi predelal tri slike v

Slika 4.11: Primer v primerjavi s sliko precej majhnega logotipa. Kljub temu, da ga je celo s prostim očesom težko opaziti (tudi na povečani sliki), ga je algoritem vseeno zaznal.

polni HD resoluciji (1920x1080). V [24] zasledimo podatek, da na GPU-ju temelječa izvedenka algoritma MSER lahko pregleda tudi do devetdeset slik polne HD resolucije na sekundo.

Poglavje 5

Sklep

V diplomski nalogi smo se ukvarjali s problemom detekcije avtomobilskih logotipov. Osredotočili smo se na avtomobilske logotipe, vendar bi bilo mogoče predlagane postopke uporabiti tudi na drugih logotipih in znakih.

Postopek je sestavljen iz odkrivanja regij z algoritmom MSER [12], iz katerih nato tvorimo hipoteze o prisotnosti logotipov v sliki. Območja v sliki za katera menimo, da bi lahko predstavljala logotip, opišemo s histogramom gradientov [11] ter z modelom naključnih gozdov [13] uvrstimo nastali vektor v enega izmed poznanih razredov. Če dovolj dreves v modelu glasuje za isti razred, privzamemo, da smo logotip našli.

Poleg algoritma smo izdelali javno dostopno anotirano zbirko fotografij avtomobilov, ki vključuje fotografije vrste v Sloveniji pogostih znamk. Zbirka je dostopna na spletnih straneh laboratorija Vicos [14].

5.1 Nadaljnje delo

Osrednji del naše metode predstavlja algoritem MSER [12], vendar pa je ta od vseh korakov tudi najpočasnejši. Paralelna izvedba tega algoritma bi zelo pohitrila celoten algoritem, vendar trenutno ne obstaja nobena javno dostopna implementacija. Zaslediti pa je mogoče omembe iz industrije [24], ki kažejo, da je bila že razvita, ter da je prinesla celo deset in večkratne pohitritve. Glede na popularnost tega algoritma, bi bila objava paralelnega postopka MSER kar pomemben prispevek. Trenutne izvedbe algoritma MSER omogočajo, da se jim poda nabor parametrov

za izbiro regij. Ker je mogoče algoritem uporabiti za več namenov na isti sliki, bi imelo smisel razširiti algoritem, da bi omogočal več naborov parametrov. Sam postopek zaradi tega ne bi bil bistveno otežen, prinesel pa bi precejšen prihranek časa.

Algoritem za zaznavanje logotipov je bil zamišljen kot del sistema za nadzor prometa, ki vključuje tudi samodejno branje registrskih tablic ter prepoznavanje modela (poleg znamke avtomobila). V tej diplomski nalogi so že bili predstavljeni nekateri elementi, kot je zaznavanje tablice, manjka še integracija v celoto.

Literatura

[1] Y. Jiang, J. Meng, J. Yuan: *Randomized visual phrases for object search.* V: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on.* IEEE, 2012, str. 3100–3107.

[2] A. Psyllos, C.-N. Anagnostopoulos, E. Kayafas: *M-SIFT: A new method for Vehicle Logo Recognition.* V: *Vehicular Electronics and Safety (ICVES), 2012 IEEE International Conference on.* IEEE, 2012, str. 261–266.

[3] A. D. Bagdanov, L. Ballan, M. Bertini, A. Del Bimbo: *Trademark matching and retrieval in sports video databases.* V: *Proceedings of the international workshop on Workshop on multimedia information retrieval.* ACM, 2007, str. 79–86.

[4] J. Greenhalgh, M. Mirmehdi: *Traffic sign recognition using MSER and Random Forests.* V: *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European.* IEEE, 2012, str. 1935–1939.

[5] D. G. Lowe: *Object recognition from local scale-invariant features.* V: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, zvezek 2. Ieee, 1999, str. 1150–1157.

[6] P. Kovesi: *Image features from phase congruency.* Videre: Journal of computer vision research **1**:3 (1999) str. 1–26.

[7] D. H. Ballard: *Generalizing the Hough transform to detect arbitrary shapes.* Pattern recognition **13**:2 (1981) str. 111–122.

[8] K. Fukushima: *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.* Biological cybernetics **36**:4 (1980) str. 193–202.

[9] S. Behnke: *Hierarchical neural networks for image interpretation.* zvezek 2766 Springer, 2003.

[10] W. Thubsaeng, A. Kawewong, K. Patanukhom: *Vehicle logo detection using convolutional neural network and pyramid of histogram of oriented gradients.* V: *Computer Science and Software Engineering (JCSSE), 2014 11th International Joint Conference on.* IEEE, 2014, str. 34–39.

[11] N. Dalal, B. Triggs: *Histograms of oriented gradients for human detection.* V: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, zvezek 1. IEEE, 2005, str. 886–893.

[12] J. Matas, O. Chum, M. Urban, T. Pajdla: *Robust wide-baseline stereo from maximally stable extremal regions.* Image and vision computing **22**:10 (2004) str. 761–767.

[13] L. Breiman: *Random forests.* Machine learning **45**:1 (2001) str. 5–32.

[14] *Zbirka slik avtomobilskih logotipov.* 2015 URL `http://www.vicos.si/Downloads/VLDS15`.

[15] R. C. Gonzalez, R. E. Woods: *Digital image processing.* 2002.

[16] H. Bay, T. Tuytelaars, L. Van Gool: *Surf: Speeded up robust features* V: *Computer Vision–ECCV 2006* Springer 2006 str. 404–417.

[17] J. Canny: *A computational approach to edge detection.* Pattern Analysis and Machine Intelligence, IEEE Transactions on :6 (1986) str. 679–698.

[18] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool: *Speeded-up robust features (SURF).* Computer vision and image understanding **110**:3 (2008) str. 346–359.

[19] M. Donoser, H. Bischof: *Efficient maximally stable extremal region (MSER) tracking.* V: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, zvezek 1. IEEE, 2006, str. 553–560.

[20] D. Nistér, H. Stewénius: *Linear time maximally stable extremal regions* V: *Computer Vision–ECCV 2008* Springer 2008 str. 183–196.

[21] S. Russell, P. Norvig, A. Intelligence: *A modern approach*. Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs **25**(1995).

[22] V. Prisacariu, I. Reid: *fastHOG-a real-time GPU implementation of HOG*. Department of Engineering Science :2310/9 (2009).

[23] *OpenCV*. 2015 URL `http://opencv.org/`, [Dostopano 1.1.2015].

[24] S. Varah, N. Grujic: *Target detection and tracking using a parallel implementation of maximally stable extremal regions*. 2013 URL `http://on-demand.gputechconf.com/gtc/2013/presentations/S3397-Target-Detection-Tracking-Parallel-MSER-Implementation.pdf`, [Dostopano 1.1.2015].

[25] P. KaewTraKulPong, R. Bowden: *An improved adaptive background mixture model for real-time tracking with shadow detection* V: *Video-Based Surveillance Systems* Springer 2002 str. 135–144.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay: *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research **12**(2011) str. 2825–2830.

[27] L. Breiman: *Technical note: Some properties of splitting criteria*. Machine Learning **24**:1 (1996) str. 41–47.